A Comparison of Three High-Precision Quadrature Schemes

David H. Bailey [a], Xiaoye S. Li [b],

[a] *Lawrence Berkeley National Laboratory, Berkeley, CA 94720.* *This work is supported by the Director, Office of Computational and Technology Research, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy, under contract number DE-AC03-76SF00098.*

[b] *Lawrence Berkeley National Laboratory, Berkeley, CA 94720*

Abstract

The authors have implemented three numerical quadrature schemes, using the new Arbitrary Precision (ARPREC) software package, with the objective of seeking a completely "automatic" arbitrary precision quadrature facility, namely one that does not rely on *a priori* information of the function to be integrated. Such a facility is required, for example, to permit the experimental identification of definite integrals based on their numerical values. The performance and accuracy of these three quadrature schemes are compared using a suite of 15 integrals, ranging from continuous, well-behaved functions on finite intervals to functions with vertical derivatives and integrable singularities at endpoints, as well as several integrals on an infinite interval.

Key words: numerical quadrature, arbitrary precision

1 Introduction

Numerical quadrature has a long and distinguished history, including contributions by Newton, who devised the basis of what is now known as the Newton-Cotes scheme, and Gauss, who devised what is now known as Gaussian quadrature. In the twentieth century, numerous additional schemes were

Email addresses: dhbailey@lbl.gov (David H. Bailey), xsli@lbl.gov (Xiaoye S. Li).

devised, including extended Simpson rules, adaptive quadrature, Romberg integration, Clenshaw-Curtis integration and others. In addition, numerous "kernels" were devised that permit these schemes to efficiently compute definite integrals of functions that include a particular expression as a factor.

Virtually all of these techniques, as well as their practical implementations on computers, have been targeted to computing definite integrals to the accuracy of 15 digits or less, namely the limits of ordinary IEEE 64-bit floating-point precision. Relatively little attention has been paid to the issues of very high precision quadrature, in part because few serious applications have been known for such techniques. The software packages *Mathematica* and *Maple* include arbitrary precision arithmetic, together with numerical integration to high precision. These facilities are generally quite good, although in many cases they either fail or require an unreasonably long run time.

In the past few years, computation of definite integrals to high precision has emerged as a useful tool in experimental mathematics. In particular, it is often possible to recognize an otherwise unknown definite integral in analytic terms, provided its numerical value can be calculated to high accuracy. Such "experimental" evaluations of integrals often involve integer relation detection, which means finding integers a_i, not all zero, such that for a given n-long real vector (x_i), we have $a_1x_1+a_2x_2+\cdots+a_nx_n = 0$. Integer relation computations are used here to determine whether the numerical value of a definite integral is given by a formula of a certain type with unknown integer or rational coefficients. The most frequently used integer relation detection algorithm is the PSLQ algorithm [3]. It and other integer relation schemes require very high precision (often hundreds or thousands of decimal digits) in both the input data and in the operation of the algorithm to obtain meaningful results.

As one example, recently one of the authors, together with Jonathan Borwein and Greg Fee of Simon Fraser University in Canada, were inspired by a recent problem in the *American Mathematical Monthly* [1]. They found by using one of the quadrature routines described in this paper, together with a PSLQ integer relation detection program, that if $C(a)$ is defined by

$$C(a) = \int_0^1 \frac{\arctan(\sqrt{x^2+a^2})\,dx}{\sqrt{x^2+a^2}(x^2+1)}$$

then

$$\begin{aligned} C(0) &= \pi \log 2/8 + G/2 \\ C(1) &= \pi/4 - \pi\sqrt{2}/2 + 3\sqrt{2}\arctan(\sqrt{2})/2 \\ C(\sqrt{2}) &= 5\pi^2/96 \end{aligned}$$

where $G = \sum_{k\geq 0}(-1)^k/(2k+1)^2$ is Catalan's constant (the third of these results is the result from the *Monthly*). These particular results then led to the following general result, among others:

$$\int_0^\infty \frac{\arctan(\sqrt{x^2+a^2})\,dx}{\sqrt{x^2+a^2}(x^2+1)} = \frac{\pi}{2\sqrt{a^2-1}}\left[2\arctan(\sqrt{a^2-1}) - \arctan(\sqrt{a^4-1})\right]$$

As another example, recently Borwein and Roland Girgensohn of Zentrum Mathematik in Germany were able to derive an analytic representation for sums of the form [7]

$$b_2(k) = \sum_{n\geq 1} \frac{n^k}{\binom{2n}{n}}$$
$$b_3(k) = \sum_{n\geq 1} \frac{n^k}{\binom{3n}{n}2^n}$$

by transforming them into the integral expressions

$$b_2(k) = \sum_j c_j^k(2) \int_0^1 [1-x(1-x)]^{-j}\,dx$$
$$b_3(k) = \sum_j c_j^k(3) \int_0^1 [1-x^2(1-x)]^{-j}\,dx$$

where the c function involves certain combinatorial values. Borwein and Girgensohn then identified these two integrals (named $d_2(k)$ and $d_3(k)$) by using a high-precision quadrature routine, similar to those described here, combined with an integer relation detection facility. In this manner they obtained $d_2(0) = 1,\ d_2(1) = 2\pi/(3\sqrt{3})$, and the recursion

$$(3p-3)d_2(p) = (7p-12)d_2(p-1) - (4p-10)d_2(p-2)$$

for $p \geq 2$. In a similar manner they obtained $d_3(0) = 1,\ d_3(1) = (3\log 2 + \pi)/5,\ d_3(2) = (90 + 96\log 2 + 37\pi)/250$, and the recursion

$$\begin{aligned} 25(p-1)(p-2)d_3(p) = {} & (77p-153)(p-2)d_3(p-1) \\ & -[79(p-2)(p-3)+p+21]d_3(p-2) \\ & +3(3p-10)(3p-8)d_3(p-3) \end{aligned}$$

for $p \geq 3$.

As a third example, recently Borwein and one of the present authors determined that

$$\int_0^{\pi/2} \mathrm{erf}^2\left(\sqrt{a}\cos x\right)\cos^2(x)\ dx$$
$$= \sum_{N=0}^{\infty} \frac{\left(\frac{-a}{4}\right)^{N+1}(8N+12)\binom{2N}{N}}{(N+2)!}\,\mathrm{F}\left(\frac{1}{2}, -N, -N-\frac{1}{2}; \frac{3}{2}, -N+\frac{1}{2}; -1\right),$$

where F denotes the hypergeometric function.

In some cases, *Maple* or *Mathematica* is able to evaluate a definite integral analytically, but the resulting expressions are extremely complicated, and thus not very useful. For example, although the integrals

$$I_1 = \int_0^1 \frac{t^2 \ln(t)\ dt}{(t^2-1)(t^4+1)}$$
$$I_2 = \int_0^{\pi/4} \frac{t^2\ dt}{\sin^2(t)}$$
$$I_3 = \int_0^{\pi} \frac{x \sin x\, dx}{1+\cos^2 x}$$

are successfully evaluated by *Maple* and *Mathematica*, the results are lengthy expressions involving advanced functions and complex entities. In the third problem, for instance, the expression produced by *Mathematica* continues for more than 100 lines. We suspect that there are considerably simpler closed-form versions of these integrals. Indeed, we can obtain the following, based solely on the high-precision numerical values of these integrals, combined with integer relation computations:

$$I_1 = \pi^2(2-\sqrt{2})/32$$
$$I_2 = -\pi^2/16 + \pi\ln(2)/4 + G$$
$$I_3 = \pi^2/4,$$

These and numerous other examples that we could cite underscore the need for a truly general-purpose, high-precision quadrature facility, by which we mean a computer program that can numerically evaluate any definite integral to high precision, given nothing other than the function definition in a separate user

subprogram. In other words, we seek a quadrature facility that does not rely on symbolic manipulation, the presence or absence of certain "kernels" in the integrand, bounds on the magnitude of the function or any of its derivatives, or any other *a priori* knowledge of the specific function to be integrated. We also seek a scheme that is well-suited to highly parallel implementation, so that a parallel computer system can be utilized when required for significantly faster run times. This latter requirement by itself rules out reliance on symbolic manipulation software, or on commercial products such as *Mathematica* and *Maple*, since these are not yet available for parallel computer systems.

The only assumptions that we grant is that the function to be integrated has a finite definite integral and is infinitely differentiable within the given interval. It may have a singularity (either a blow-up singularity or a vertical derivative) at one or both endpoints. The interval itself may be finite, semi-infinite or the entire real line. Note that definite integrals of functions with a finite set of discontinuities or other singularities within an interval may be computed as a sum of definite integrals on subintervals, so that the assumption given above encompasses a broad range of functions of interest.

2 The ARPREC Software

The quadrature techniques we describe below have been implemented using the new Arbitrary Precision (ARPREC) computation package [4]. This software is based in part on the Fortran-90 MPFUN package [6], which in turn is based on an earlier Fortran-77 package [5]. In the Fortran-90 version of MPFUN, object-oriented facilities built into the Fortran-90 language (notably custom datatypes and operator overloading) were exploited to permit Fortran programmers to utilize the MPFUN library by making only minor changes to the user's source code.

The ARPREC library extends the functionality of the MPFUN packages to the realm of C/C++ programs. In particular, the ARPREC package combines the following features, which we believe to be unique for currently available software of this type:

- Code written in C++ for high performance and broad portability.
- Both C++ and Fortran-90 translation modules, which permit conventional C++ and Fortran-90 programs to utilize the package with only very minor changes to source code.
- Arbitrary precision integer, floating and complex datatypes.
- Support for datatypes with differing precision levels.
- Inter-operability with conventional integer and floating-point datatypes.
- Many common transcendental functions (sqrt, exp, sin, erf, etc).

- Quadrature routines (for numerical integration).
- PSLQ routines (for integer relation detection).
- Polynomial equation solutions, both real and complex roots.
- Special routines, including FFT-based multiplication, for extra-high precision (> 1000 digits) computation.

3 The Three Quadrature Schemes

We describe here three state-of-the-art numerical quadrature schemes that are suitable for computing definite integrals to very high precision. We considered other quadrature schemes, including several of the classical schemes mentioned in the introduction, but we found that they are not competitive with these three schemes, particularly when high-precision results are required.

- *QUADGS: A Gaussian quadrature scheme.* Gaussian quadrature certainly is not new, although most descriptions in the literature do not address the requirements of arbitrary precision implementation. This scheme approximates an integral on $[-1, 1]$ as the sum $\sum_{0 \le j < n} w_j f(x_j)$, where the abscissas x_j are the roots of the n-th degree Legendre polynomial $P_n(x)$ on $[-1, 1]$, and the weights x_j are

$$w_j = \frac{-2}{(n+1)P'_n(x_j)P_{n+1}(x_j)}$$

[2, pg. 187]. We compute the abscissas using a Newton iteration scheme with a dynamically changing level of precision (approximately doubling with each iteration), so that the total cost is only about twice times the cost of the final iteration. The starting value for x_j in these iterations is given by $\cos[\pi(j - 1/4)/(n+1/2)]$ [12, pg. 125]. We compute the Legendre polynomial function values using an n-long iteration of the recurrence $P_0(x) = 0,\ P_1(x) = 1$ and

$$(k+1)P_{k+1}(x) = (2k+1)xP_k(x) - kP_{k-1}(x)$$

for $k \ge 2$. The derivative is computed as $P'_n(x) = n(xP_n(x) - P_{n-1}(x))/(x^2 - 1)$. Multiple levels (in other words, multiple sets of abscissas and weights) are typically pre-computed, where each level has twice as many abscissas and weights as the level before. When evaluating an integral, we start with the first level, and continue with additional levels until we either exhaust our set of pre-computed abscissas and weights, or else we are satisfied with the accuracy of our result (see Section 5).

The number n of abscissas and weights required at level k is $n = 3 \cdot 2^k$ in the implementation below, so that the total required for m levels is $\sum_{k \le m} 3 \cdot 2^k \approx 6 \cdot 2^m$. The cost of computing abscissas and weights at a given level with this scheme increases quadratically with n. The abscissas and weights

can alternately be computed using an eigenvector scheme due to Golub and Welch [10], although the cost for this method also increases quadratically with n.

- *QUADERF: A error function-based quadrature scheme.* This program approximates an integral on $[-1, 1]$ as a sum $\sum_{0 \le i < n} w_i f(x_i)$, as with Gaussian quadrature, but here the abscissas x_j are given by $\mathrm{erf}(hj)$, where $\mathrm{erf}(x) = (2/\sqrt{\pi}) \int_0^x e^{-t^2}\ dt$, and the weights are given by $(2/\sqrt{\pi})e^{-(hj)^2}$. We compute the error function $\mathrm{erf}(x)$ as $1 - \mathrm{erfc}(x)$, using the following formula given by Crandall [9, pg. 85] (who in turn attributes it to a 1968 paper by Chiarella and Reichel [8]):

$$\mathrm{erfc}(t) = \frac{e^{-t^2}\alpha t}{\pi}\left(\frac{1}{t^2} + 2\sum_{k\ge 1}\frac{e^{-k^2\alpha^2}}{k^2\alpha^2 + t^2}\right) + \frac{2}{1 - e^{2\pi t/\alpha}} + E$$

  where $|E| < e^{-\pi^2/\alpha^2}$. The parameter α is chosen large enough to ensure that the error E is sufficiently small. As with the Gaussian scheme, multiple "levels" of abscissas and weights are typically pre-computed, with each level having approximately twice as many abscissas and weights as the level before. Here level k uses $h = 2^{2-k}$. Note that in this scheme, the even-indexed abscissas and weights at one level are merely the full set of abscissas and weights at the previous level. Thus only the odd-indexed half of the abscissas and weights need to be computed at each level (after the first level), and the function to be integrated needs to be evaluated only at the odd-indexed abscissas at each level.

  The number n of abscissas and weights required at level k depends on the numeric precision being used. In the implementation below, $n = 4 \cdot 2^k$ approximately, so that the total required for m levels is approximately $\sum_{k \le m} 4 \cdot 2^k \approx 8 \cdot 2^m$. The cost of computing abscissas and weights with this scheme increases only linearly with n.
- *QUADTS: A tanh-sinh quadrature scheme.* This scheme is similar to the error function scheme. In this case the abscissas are chosen as $x_j = \tanh(\pi/2 \cdot \sinh(hj))$ and the weights $w_j = \pi/2 \cdot \cosh(hj)/\cosh^2(\pi/2 \cdot \sinh(hj))$. In this case level k uses $h = 2^{-k}$. As with the QUADERF scheme, the even-indexed abscissas and weights at one level are merely the full set of abscissas and weights at the previous level. This scheme was first introduced by Takahasi and Mori [13,11].

  The number n of abscissas and weights required at level k depends on the numeric precision being used. In the implementation here, $n = 3.3 \cdot 2^k$ approximately, so that the total required for m levels is approximately $\sum_{k \le m} 3.3 \cdot 2^k \approx 6.6 \cdot 2^m$. The cost of computing abscissas and weights with this scheme increases only linearly with n.

4 The Euler-MacLaurin Summation Formula

The error function and tanh-sinh quadrature schemes are based on the Euler-MacLaurin summation formula, which can be stated as follows [2, pg. 180]. Let $m \geq 0$ and $n \geq 1$ be integers, and define $h = (b-a)/n$ and $x_j = a+jh$ for $0 \leq j \leq n$. Further assume that the function $f(x)$ is at least $(2m+2)$-times continuously differentiable on $[a, b]$. Then

$$\int_a^b f(x)\,dx = h\sum_{j=0}^{n} f(x_j) - \frac{h}{2}\left(f(a)+f(b)\right) - \sum_{i=1}^{m} \frac{h^{2i}B_{2i}}{(2i)!}\left(f^{(2i-1)}(b) - f^{(2i-1)}(a)\right) - E$$

where B_{2i} denote the Bernoulli numbers, and

$$E = \frac{h^{2m+2}(b-a)B_{2m+2}f^{2m+2}(\xi)}{(2m+2)!}$$

for some $\xi \in (a, b)$.

In the circumstance where the function $f(x)$ and all of its derivatives are zero at the endpoints a and b (as in a smooth, bell-shaped function), the second and third terms of the Euler-MacLaurin formula are zero. Thus the error in a simple step-function approximation to the integral, with interval h, is simply E. But since E is then less than a constant times $h^{2m+2}/(2m+2)!$, for any m, we conclude that the error goes to zero more rapidly than any power of h. In the case of a function defined on $(-\infty, \infty)$, the Euler-MacLaurin summation formula still applies to the resulting infinite sum approximation, provided as before that the function and all of its derivatives tend to zero for large arguments.

This principle is utilized in the error function and tanh-sinh schemes by transforming the integral of $f(x)$ on the interval $(-1, 1)$ to an integral on $(-\infty, \infty)$ using the change of variable $x = g(t)$. Here $g(x)$ is some monotonic function with the property that $g(x) \to 1$ as $x \to \infty$ and $g(x) \to -1$ as $x \to -\infty$, and also with the property that $g'(x)$ and all higher derivatives rapidly approach zero for large positive and negative arguments. In this case we can write, for $h > 0$,

$$\int_{-1}^{1} f(x)\,dx = \int_{-\infty}^{\infty} f(g(t))g'(t)\,dt \;=\; h\sum_{-\infty}^{\infty} w_j f(x_j)$$

where $x_j = g(hj)$ and $w_j = g'(hj)$. If the convergence of $g'(t)$ and its derivatives to zero is sufficiently rapid for large $|t|$, then even in cases where $f(x)$ has a vertical derivative or an integrable singularity at one or both endpoints, the resulting integrand $f(g(t))g'(t)$ will be a smooth bell-shaped function for which the Euler-Maclaurin summation formula applies. In such cases the error in the above approximation decreases faster than any power of h.

The error function integration scheme uses $g(t) = \mathrm{erf}(t)$ and $g'(t) = (2/\sqrt{\pi})e^{-t^2}$. Note that $g'(t)$ is merely the bell-shaped probability density function, which is well-known to converge rapidly to zero, together with all of its derivatives, for large arguments. The tanh-sinh scheme uses $g(t) = \tanh(\pi/2 \cdot \sinh t)$ and $g'(t) = \pi/2 \cdot \sinh t / \cosh^2(\pi/2 \cdot \sinh t)$, for which the convergence to zero is compound exponential, even faster than the probability density function.

In practice, for functions that are bounded and well behaved on a finite closed interval, all three of these numerical integration schemes exhibit quadratic convergence: after a few initial levels, subsequent levels produce approximations with approximately twice the number of correct digits as the previous level. Further, as we shall see, the error function and tanh-sinh schemes also exhibit quadratic convergence even for many functions with vertical derivatives or singularities at one or both endpoints of the interval.

5 Error Estimation

As mentioned above, we seek a general purpose high-precision numerical integration facility that does not depend on any *a priori* knowledge of the function or its derivatives. Thus the theoretical bounds that are known for many quadrature schemes are not of much use here. Instead, we use the following error estimation scheme, which is inspired by (although it does not critically rely on) the quadratically convergent behavior normally achieved by the above schemes.

Let S_k be the computed approximations of the integral for levels k up to level n. Then the estimated error E_n at level n is one if $n \leq 2$, zero if $S_n = S_{n-1}$, and otherwise 10^d, where $d = \min[0, \max(d_1^2/d_2,\ 2d_1,\ d_3)]$, with $d_1 = \log_{10}|S_n - S_{n-1}|$, $d_2 = \log_{10}|S_n - S_{n-2}|$, and $d_3 = \log_{10}(\epsilon \cdot \max_j |f(x_j)|)$. In the definition of d_3 in the previous sentence, ϵ is the "machine epsilon" of the multiprecision system being used (in the examples below, $\epsilon = 10^{-412}$), and the maximum indicated in this expression is taken over all abscissas x_j at the n-th level.

6 Test Problems

The following set of 15 integrals were used as a test suite to compare these three quadrature schemes. In each case an analytic result is known, as shown below, facilitating the checking of results. The 15 integrals are listed in 5 groups:

- 1–4: Continuous functions on finite intervals.
- 5–6: Continuous functions on finite intervals, but with a vertical derivative at an endpoint.
- 7–10: Functions on finite intervals with an integrable singularity at an endpoint.
- 11–13: Functions on an infinite interval.
- 14–15: Oscillatory functions on an infinite interval.

$$1: \int_0^1 t \log(1+t)\, dt \;=\; 1/4$$

$$2: \int_0^1 t^2 \arctan t\, dt \;=\; (\pi - 2 + 2\log 2)/12$$

$$3: \int_0^{\pi/2} e^t \cos t\, dt \;=\; (e^{\pi/2} - 1)/2$$

$$4: \int_0^1 \frac{\arctan(\sqrt{2+t^2})}{(1+t^2)\sqrt{2+t^2}}\, dt \;=\; 5\pi^2/96$$

$$5: \int_0^1 \sqrt{t} \log t\, dt \;=\; -4/9$$

$$6: \int_0^1 \sqrt{1-t^2}\, dt \;=\; \pi/4$$

$$7: \int_0^1 \frac{t}{\sqrt{1-t^2}} dt \;=\; 1$$

$$8: \int_0^1 \log t^2\, dt \;=\; 2$$

$$9: \int_0^{\pi/2} \log(\cos t)\, dt \;=\; -\pi \log(2)/2$$

$$10: \int_0^{\pi/2} \sqrt{\tan t}\, dt \;=\; \pi\sqrt{2}/2$$

$$11: \int_0^{\infty} \frac{1}{1+t^2}\, dt \;=\; \int_0^{1} \frac{ds}{1-2s+2s^2} \;=\; \pi/2$$

$$12: \int_0^{\infty} \frac{e^{-t}}{\sqrt{t}}\, dt \;=\; \int_0^{1} \frac{e^{1-1/s}\, ds}{\sqrt{s^3-s^4}} \;=\; \sqrt{\pi}$$

$$13: \int_0^{\infty} e^{-t^2/2}\, dt \;=\; \int_0^{1} \frac{e^{-(1/s-1)^2/2}\, ds}{s^2} \;=\; \sqrt{\pi/2}$$

$$14: \int_0^{\infty} e^{-t} \cos t\, dt \;=\; \int_0^{1} \frac{e^{1-1/s} \cos(1/s-1)\, ds}{s^2} \;=\; 1/2$$

$$15: \int_0^{\infty} \frac{\sin t}{t}\, dt \;=\; \int_0^{\pi} \frac{\sin t}{t}\, dt + 40320 \int_0^{1/\pi} t^7 \sin(1/t)\, dt - \frac{1}{\pi} + \frac{2}{\pi^3} - \frac{24}{\pi^5} + \frac{720}{\pi^7}$$
$$= \frac{\pi}{2}$$

Problem 4, as was mentioned above, appeared in Sept. 2002 *American Mathematical Monthly* [1]. The others were constructed by the authors. Problems 11-15 are integrals on an infinite interval, which is in each case here $[0, \infty)$. Except for Problem 15, such integrals are evaluated by using the variable transformation $t = 1/s - 1$, as shown above. In Problem 15, the integral is written as the sum of integrals on $[0, \pi]$ and $[\pi, \infty)$. Then integration by parts is applied several times to the second integral of this pair, resulting in the expression shown above. This expression requires the evaluation of the integrals $\int_0^{\pi} t^{-1} \sin t\, dt$ and $\int_0^{1/\pi} t^7 \sin(1/t)\, dt$, which are significantly better behaved than the original, resulting in faster convergence. Even with this transformation, however, problem 15 remains the most difficult of the entire set.

7 Results of Tests

The three quadrature routines were each implemented using the ARPREC arbitrary precision computation package [4], in a virtually identical programming style, with the user working precision set at 400 digits (the actual working precision employed by the software is slightly higher, roughly 412 digits). We seek answers good to a target accuracy of 10^{-390}, making allowance for some numerical round-off error. In each case, nine levels of abscissas and weights were pre-computed. Each quadrature routine was then run blindly—beginning at level one and continuing at successively higher levels (each of which approx-

	QUADGS			QUADERF			QUADTS		
Prob.	Level	Time	Error	Level	Time	Error	Level	Time	Error
Init	9	2778.29		9	131.80		9	45.46	
1	6	8.72	10^{-406}	9	57.43	10^{-406}	7	13.69	10^{-390}
2	6	8.86	10^{-406}	9	36.17	10^{-406}	8	21.86	10^{-406}
3	5	4.16	10^{-405}	9	44.06	10^{-405}	7	12.01	10^{-405}
4	6	8.78	10^{-405}	9	92.48	10^{-406}	8	38.43	10^{-406}
5	9	78.00	10^{-11}	9	68.15	10^{-406}	7	16.08	10^{-406}
6	9	3.65	10^{-12}	9	3.94	10^{-406}	7	0.90	10^{-392}
7	9	4.39	10^{-4}	8	2.39	10^{-210}	6	0.55	10^{-196}
8	9	75.84	10^{-6}	9	65.58	10^{-405}	7	15.29	10^{-400}
9	9	99.83	10^{-7}	9	69.96	10^{-403}	7	17.97	10^{-390}
10	9	31.68	10^{-4}	8	7.49	10^{-203}	6	2.36	10^{-194}
11	7	0.61	10^{-405}	9	3.04	10^{-255}	9	2.59	0
12	9	47.55	10^{-4}	9	18.88	10^{-132}	9	26.09	10^{-203}
13	9	41.74	10^{-353}	9	11.30	10^{-91}	9	17.97	10^{-241}
14	9	71.69	10^{-126}	9	26.87	10^{-68}	9	44.39	10^{-165}
15	5/9	34.45	10^{-19}	9/9	41.14	10^{-17}	7/9	31.66	10^{-19}

imately doubles the run time) until one of these three conditions was met: (1) the maximum level (level nine) is completed; (2) the estimated error achieves the accuracy target (10^{-390}); or (3) the absolute values of the function near an endpoint are found to be so large that even when multiplied by very small weights, the resulting summands are not sufficiently small to insure full accuracy in the result. In the last case, the usage of additional levels of abscissas and weights will not further improve the accuracy—higher numeric precision is required to attain the target accuracy.

The results of these tests are given in the table below. The first line gives the run time, in seconds, for the initialization process. Altogether, the nine levels of initialization produced 3066 multi-precision abscissas (and the same number of weights) for the Gaussian scheme, 4033 for the error function scheme, and 3305 for the tanh-sinh scheme. In problems 15, where a two-step scheme is used, the number of levels used for both steps are shown in the table. Note that in problem 11, the tanh-sinh quadrature routine produced a value that was identical to the reference value (i.e. no error whatsoever).

8 Analysis

The Gaussian quadrature routine was clearly superior for the first set of problems, namely integrals of bounded, well-behaved continuous functions on finite intervals. It was as much as ten times faster than the error function routine on these problems, and as much as four times faster than the tanh-sinh routine. Its accuracy on these problems was consistently "machine epsilon." The Gaussian routine also did quite well on problems 11 and 13. But for the other problems, which are characterized by functions that are not well-behaved at endpoints, its accuracy was quite poor, even when all nine levels of abscissas and weights were utilized. Another drawback of the Gaussian scheme is that its initialization time is 20 times longer than the error function scheme and 57 times longer than the tanh-sinh routine. This is due to the fact that the amount of computation in the Gaussian initialization scales as n^2, where n is the number of abscissas and weights, whereas the scaling is linear with the other two schemes. The authors are not aware of any solution to this feature of Gaussian quadrature—all known schemes for computing abscissas and weights have n^2 scaling.

The error function quadrature routine was several times slower than Gaussian quadrature on the well-behaved problems, as noted above, but it produced highly accurate answers on almost all problems. In cases where it did not achieve 400 digits accuracy, it is clear that it could do better, provided additional levels of abscissas and weights are used, and a higher working precision is employed. It did poorly on problem 15, but so did the other two routines. For problems seven and ten, the error function scheme stopped before using all nine levels of abscissas and weights, even though it has achieved only about 200 digits accuracy, because it encountered case 3 of the termination criteria above: function values near one of the endpoints became very large, so that as a result higher levels of abscissas and weights will not further increase accuracy. On these problems, roughly 800 digit working precision is required to achieve 400 digit accuracy in the result.

The tanh-sinh quadrature routine run times were faster than the error function routine in most cases, although not as fast as the Gaussian quadrature routine on the well-behaved problems. Its initialization time is easily the fastest of the three. It did poorly on problem 15, but no worse than the other two routines. As with the error function routine, the tanh-sinh function scheme achieved only about 200 digits accuracy on problems seven and ten, because the function values near one of the endpoints are very large. For these problems, roughly 800 digit working precision is required to achieve a full 400 digit precision in the result, as noted above with the error function scheme.

One additional item of note here is that the error estimation scheme given in

Section 5 performed very well in these tests, for each of the three quadrature routines. In most cases it produced a value that was either identical to (in power of ten), or a few orders of magnitude higher than, the actual error in the final result. In no case did it underestimate the actual error of the final result, except for problem 15, where it was roughly four orders of magnitude too high for each of the three routines. But this is easily explained, since the second of the two integrals involved here is multiplied by the coefficient 40,320 in the main program outside the integration routine (see the description of problem 15 in section four above). In other words, even in problem 15, the estimated error was accurate for the actual definite integrals that were calculated.

9 Summary

Each of these quadrature routines has proven its value in a certain domain of quadrature problems. Overall, the tanh-sinh scheme appears to be the best. It combines uniformly excellent accuracy (except for problem 15) with fast run times, typically only a few seconds. It is the nearest we have to a truly all-purpose quadrature scheme at the present time. These three programs, as well as the associated ARPREC arbitrary precision computation software, are available from the website

`http://www.nersc.gov/~dhbailey/mpdist`

We wish to add here that each of the three schemes described above are well-suited for parallel computation. The present authors are thus working on an implementation for highly parallel computer platforms. The implementation is based on the MPI programming model. We expect reasonably linear scalability up to roughly 256 or possibly more processors.

References

[1] Zafar Ahmed, "Definitely an Integral," *American Mathematical Monthly*, vol. 109 (2002), no. 7, pg. 670–671.

[2] Kendall E. Atkinson, *Elementary Numerical Analysis*, John Wiley and Sons, 1993.

[3] David H. Bailey and David Broadhurst, "Parallel Integer Relation Detection: Techniques and Applications," to appear in *Mathematics of Computation*, available from the URL
`http://www.nersc.gov/~dhbailey/dhbpapers`

[4] David H. Bailey, Yozo Hida, Xiaoye S. Li and Brandon Thompson, "ARPREC: An Arbitrary Precision Computation Package," software and documentation

available from the URL
http://www.nersc.gov/~dhbailey/mpdist

[5] David H. Bailey, "Multiprecision Translation and Execution of Fortran Programs," *ACM Transactions on Mathematical Software*, vol. 19 (1993), pg. 288–319.

[6] David H. Bailey, "A Fortran-90 Based Multiprecision System," *ACM Transactions on Mathematical Software*, vol. 21 (1995), pg. 379–387.

[7] Jonathan M. Borwein and Roland Girgensohn, "Evaluations of Binomial Series," manuscript, 2002, available from http://www.cecm.sfu.ca/preprints/2002pp.html.

[8] C. Chiarella and A. Reichel, "On the Evaluation of Integrals Related to the Error Function," *Mathematics of Computation*, vol. 22 (1968), pg. 137–143.

[9] Richard E. Crandall, *Topics in Advanced Scientific Computation*, Springer-Verlag, 1996.

[10] G. H. Golub and J. H. Welsch, "Calculation of Gauss Quadrature Rules," *Mathematics of Computation*, vol. 22 (1969), pg. 221–230.

[11] M. Mori, "Developments in the Double Exponential Formula for Numerical Integration," *Proceedings of the International Congress of Mathematicians*, Springer-Verlag, 1991, pg. 1585–1594.

[12] William H. Press, Brian P. Flannery, Saul A. Teukolsky and William T. Vetterling, *Numerical Recipes*, Cambridge University Press, 1986.

[13] H. Takahasi and M. Mori, "Double Exponential Formulas for Numerical Integration," Publications of RIMS, Kyoto University, vol. 9 (1974), pg. 721–741.